

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-311839

(43)Date of publication of application : 02.12.1997

(51)Int.Cl.

G06F 15/16
G06F 13/14

(21)Application number : 08-126083

(22)Date of filing : 21.05.1996

(71)Applicant : HITACHI LTD

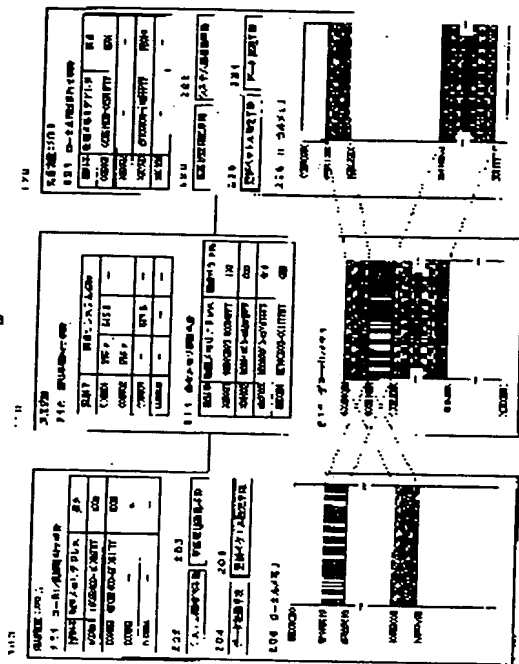
(72)Inventor : OKUHARA SUSUMU
MORIJIMA HIROSHI
MAEDA SHINGO
TAMAKI KIKUKO

(54) DATA SHARING SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To easily access shared data.

SOLUTION: The inside of a shared global memory 212 is divided into fixed units to manage each unit divided with a unique identifier in view from processors 100, 120 and a shared device 110. When an inter-system excluding means 202 in the processor 100 designates the area of an optional address space in the processor 100 with an identifier to secure exclusion, an updating state confirming means 203 checks the existence of the updating of data within a corresponding unit in the shared device 110. When data is updated, a data transferring means 204 transfers data in corresponding unit in the shared device 110 to an address space. When data in the address space is updated at the time of releasing exclusion, the data sharing system transferring means 204 transfers the data to inside of a corresponding unit in the shared device 110.



LEGAL STATUS

[Date of request for examination]

10.08.1999

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-311839

(43) 公開日 平成9年(1997)12月2日

(51) Int.Cl. ⁸	識別記号	片内整理番号	F I	技術表示箇所
G 0 6 F 15/16	3 7 0		G 0 6 F 15/16	3 7 0 M
13/14	3 1 0		13/14	3 1 0 Y

審査請求 未請求 請求項の数 3 O L (全 13 頁)

(21) 出願番号 特願平8-126083

(22) 出願日 平成8年(1996)5月21日

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 奥原 進

神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作所ソフトウェア開発本部内

(72) 発明者 守島 浩

神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作所ソフトウェア開発本部内

(72) 発明者 前田 新吾

神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作所ソフトウェア開発本部内

(74) 代理人 弁理士 春日 譲

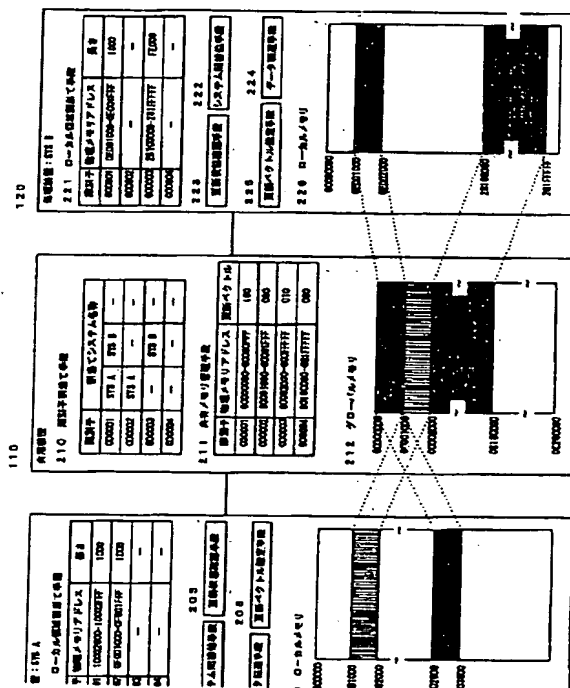
最終頁に続く

(54) 【発明の名称】 データ共用方式

(57) 【要約】 (修正有)

【課題】 容易に共用データをアクセスする。

【解決手段】 共用されたグローバルメモリ212内は、一定の単位で分割し、処理装置100、120及び共用装置110から見て一意の識別子でもって分割されたそれぞれの単位を管理する。処理装置100のシステム間排他手段202が、処理装置100内の任意のアドレス空間の領域を識別子でもって指定して排他確保する場合は、更新状態確認手段203は、共用装置110内の対応する単位内のデータの更新の有無をチェックし、更新されている時には、データ転送手段204は、共用装置110内の対応する単位内のデータをアドレス空間に転送する。排他解除する場合アドレス空間内のデータが更新されているときには、データ共用方式転送手段204は、そのデータを共用装置110内の対応する単位内に転送する。



【特許請求の範囲】

【請求項1】 複数のコンピュータシステム間で共通に使用するデータを共用装置内の共用メモリにおいて共用するデータ共用方式において、

上記共用メモリ内を一定の単位で分割し、上記複数のコンピュータシステム及び上記共用装置から見て一意の識別子をもって上記分割されたそれぞれの単位を管理し、上記コンピュータシステム内の任意のアドレス空間の領域を上記識別子をもって指定して排他確保する時には、上記共用メモリ内の対応する単位内のデータの更新の有無をチェックし、更新されている時には、上記共用メモリ内の対応する単位内のデータを上記アドレス空間に転送し、

排他解除する時には、上記アドレス空間内のデータが更新されている時には、そのデータを上記共用メモリ内の対応する単位内に転送することを特徴とするデータ共用方式。

【請求項2】 請求項1記載のデータ共用方式において、

上記コンピュータシステムは、複数の上記識別子をグループ化してアクセスするとともに、上記共用装置に対する排他確保を、グループ化された識別子の内の代表する識別子をもって行うことを特徴とするデータ共用方式。

【請求項3】 請求項1記載のデータ共用方式において、

上記コンピュータシステムが、上記コンピュータシステム上でデータを参照・更新するアクセス時期と、上記共用装置上のデータを上記コンピュータシステムに反映させるアクセス時期とを非同期とすることを特徴とするデータ共用方式。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、データ共用方式に係り、特に、複数のコンピュータシステム間で共通に使用するデータを共用装置において共用するデータ共用方式に関する。

【0002】

【従来の技術】 従来、複数のコンピュータシステム間でアドレス空間上のデータを共用する場合は、共用されている入出力装置を占有し、あらかじめ決められた位置に一旦データを書き込み後、占有状態を解除している。その後、他システムがそのデータを利用する際には、同様にして、この入出力装置を占有してから、決められた位置のデータを読み込み、アドレス空間上にあらかじめ確保した領域にデータを転送後、占有状態を解除するようにしている。即ち、個々のコンピュータシステムが、一旦、入出力装置を占有して、データの読み出しや書き込みを行い、その後、占有を解除している。ここで、複数システム間で共有する入出力装置としては、例えば、磁

装置などがある。

【0003】 なお、独立型拡張記憶装置による複数システム間でのデータ共用方法については、例えば、(株)日立製作所発行のマニュアル「プログラムプロダクトV OS 3/F Sセンタ運営-J S S 3編-」、平成7年10月発行のp. 110-117、「拡張記憶装置」の項に記載されている。

【0004】

【発明が解決しようとする課題】 しかしながら、従来のシステムにおいては、共用装置である入出力装置を占有するため、入出力装置の排他オーバーヘッドが大きく、比較的小さいデータを頻繁にアクセスすると、排他オーバーヘッドやアクセスオーバーヘッドが無視できなくなるという問題があった。

【0005】 本発明の目的は、各コンピュータシステム上で稼働するプログラムが共用装置の存在を意識せずに、自システム内の領域をアクセスするのと同様な手順により、容易にシステム間の共用データをアクセスできるデータ共用方式を提供することにある。

【0006】

【課題を解決するための手段】 上記目的を達成するために、本発明は、複数のコンピュータシステム間で共通に使用するデータを共用装置内の共用メモリにおいて共用するデータ共用方式において、上記共用メモリ内を一定の単位で分割し、上記複数のコンピュータシステム及び上記共用装置から見て一意の識別子をもって上記分割されたそれぞれの単位を管理し、上記コンピュータシステム内の任意のアドレス空間の領域を上記識別子をもって指定して排他確保する時には、上記共用メモリ内の対応する単位内のデータの更新の有無をチェックし、更新されている時には、上記共用メモリ内の対応する単位内のデータを上記アドレス空間に転送し、排他解除する時には、上記アドレス空間内のデータが更新されている時には、そのデータを上記共用メモリ内の対応する単位内に転送するようにしたものであり、かかる方式により、各コンピュータシステム上で稼働するプログラムは共用装置の存在を意識せずに、自システム内の領域をアクセスするのと同様な手順により、容易にシステム間の共用データをアクセスし得るものとなる。

【0007】 上記データ共用方式において、好ましくは、上記コンピュータシステムは、複数の上記識別子をグループ化してアクセスするとともに、上記共用装置に対する排他確保を、グループ化された識別子の内の代表する識別子をもって行うようにしたものであり、かかる方式により、排他オーバーヘッドを削減し得るものとなる。

【0008】 上記データ共用方式において、好ましくは、上記コンピュータシステムが、上記コンピュータシステム上でデータを参照・更新するアクセス時期と、上

映させるアクセス時期とを非同期とするようにしたものであり、かかる方式により、共用装置へのオーバーヘッドを低減し得るものとなる。

【0009】

【発明の実施の形態】以下、図1～図3を用いて、本発明の一実施形態によるデータ共用方式について説明する。図1は、本発明の一実施形態によるデータ共用方式を実現する複合コンピュータシステムの構成図である。

【0010】処理装置100は、共用装置110上の識別子で管理されるグローバルメモリ212と1:1で対応する自システム内のローカルメモリ206のアドレス空間上のメモリ領域を管理するローカル領域割当て手段201を備えている。ローカル領域割当て手段201は、「識別子」、「物理メモリアドレス」及び「長さ」の各項目からなるテーブル形式で、メモリ領域を管理する。

【0011】ローカル領域割当て手段201の中には、例えば、ローカルメモリ206のアドレス"10002000" Xからアドレス"10002FFF" Xまでの長さ"1000" Xの領域を「識別子」="000001"で管理し、ローカルメモリ206のアドレス"0F001000" Xからアドレス"0F001FFF" Xまでの長さ"1000" Xの領域を「識別子」="000002"で管理しているものとする。

【0012】「識別子」="000001"、「識別子」="000002"で管理されるメモリ領域は、ローカルメモリ206に図示した領域である。

【0013】さらに、処理装置100は、共有装置110へのデータ転送時に他システムと排他を確保・解放するためのシステム間排他手段202と、共有装置110上のデータが他システムによって更新されているかどうかチェックする更新状態確認手段203と、処理装置100上のローカルメモリ206から共用装置110上のグローバルメモリ212へ、または、共用装置110上のグローバルメモリ212から処理装置100上のローカルメモリ206へデータを転送するデータ転送手段204と、共用装置110上の他系の更新状態を記録する更新ベクトルの設定を行う更新ベクトル設定手段205と、自システム上のアドレス空間で参照できるローカルメモリ206とを備えている。

【0014】処理装置120は、処理装置100と同様な構成となっており、ローカル領域割当て手段221と、システム間排他手段222と、更新状態確認手段223と、データ転送手段224と、更新ベクトル設定手段225と、ローカルメモリ226とを備えている。これらの各手段は、いずれも、処理装置100における対応する手段と同一の機能を有しており、その説明については、省略する。

【0015】なお、ローカル領域割当て手段221の中

001000" Xからアドレス"0E001FFF" Xまでの長さ"1000" Xの領域を「識別子」="000001"で管理し、ローカルメモリ226のアドレス"28100000" Xからアドレス"281FFFFF" Xまでの長さ"FE000" Xの領域を「識別子」="000003"で管理していることが記録されている。

【0016】「識別子」="000001"、「識別子」="000003"で管理されるメモリ領域は、ローカルメモリ226に図示した領域である。

【0017】共用装置110は、処理装置(SYS A)100及び処理装置(SYS B)120に接続されており、その内部にメモリを保持し、接続されている処理装置100、120からの指示に従い、メモリ上のデータを操作する機能を有している。共用装置110は、例えば、磁気ディスク装置、半導体ディスク装置や独立形拡張記憶装置などによって構成される。

【0018】共用装置110は、共用装置内のメモリを分割して管理する手段として、それぞれの識別子がどのシステムに割り当てられているかの情報を記録する識別子割当て手段210と、グローバルメモリ212のアドレス空間上のメモリ領域を識別子で管理する共有メモリ管理手段211とを備え、さらに、実際のデータを配置する場所としてグローバルメモリ212を備えている。

【0019】識別子割当て手段210は、「識別子」、「割当てシステム名称」の各項目からなるテーブル形式で、識別子の割当てを管理している。識別子割当て手段210の中には、例えば、「識別子」="000001"は、処理装置100である"SYS A"、処理装置120である"SYS B"に割り当てられており、「識別子」="000002"は、"SYS A"に割り当てられており、「識別子」="000003"は、"SYS B"に割り当てられていることが記録されている。

【0020】また、共有メモリ管理手段211は、「識別子」、「物理メモリアドレス」、「更新ベクトル」の各項目からなるテーブル形式で、識別子に対するメモリの割当てを管理している。共有メモリ管理手段211の中には、例えば、グローバルメモリ212のアドレス"00000000" Xからアドレス"00000FFF" Xまでの領域を「識別子」="000001"で管理し、「更新ベクトル」が"100"であることが記録されている。「更新ベクトル」の詳細については、後述するが、3ビット構成の「更新ベクトル」は、識別子割当て手段210の「割当てシステム名称」の中の3つの欄に対応してフラグを立てるようにしている。即ち、「識別子」="000001"に対する「更新ベクトル」="100"の内、「1」が"SYS A"に対応し、「0」が"SYS B"に対応し、「0」が"ー"（割当シ

A”に対してビットが立っている(“1”)ことから、「識別子」＝“000001”で管理されるメモリ領域のデータは、処理装置100(SYS A)によって更新されていることを表している。

【0021】同様にして、共有メモリ管理手段211の中には、グローバルメモリ212のアドレス“00001000”Xからアドレス“00001FFF”Xまでの領域を「識別子」＝“000002”で管理し、

「更新ベクトル」が“000”であり、いずれの処理装置からも更新されておらず、グローバルメモリ212のアドレス“00002000”Xからアドレス“000FFFFF”Xまでの領域を「識別子」＝“000003”で管理し、「更新ベクトル」が“010”であり、処理装置120(SYS B)によって更新されており、グローバルメモリ212のアドレス“00100000”Xからアドレス“001FFFFFF”Xまでの領域を「識別子」＝“000004”で管理し、「更新ベクトル」が“000”であり、いずれの処理装置からも更新されていないことが記録されている。

【0022】「識別子」＝“000001”，…，「識別子」＝“000004”で管理されるメモリ領域は、グローバルメモリ212に図示した領域である。

【0023】次に、図2を用いて、複数システム間で定義したシステム間の共通メモリ領域をロック要求時に参照するときの手順について説明する。図2は、本発明の一実施形態によるデータ共用方式におけるロック要求処理の手順を示すフローチャートである。

【0024】以下に、図2に示した各ステップに従って、図1を参照しながら、ロック要求処理について説明するが、ここで、図1における識別子割当て手段210の「割当てシステム」の欄にはまだ何も記録されておらず、また、共有メモリ管理手段211の「更新ベクトル」の欄はいずれも“000”となっているものとする。

【0025】図2のステップ1において、処理装置100上のプログラムは、予め確保してあるアドレス空間上のローカルメモリ206の「アドレス」＝“10002000”Xから「長さ」＝“1000”X分の領域を、システム間の共通メモリとしてアクセスするために、処理装置120上のプログラムとあらかじめ決めておいた「識別子」＝“000001”によりロック要求を発行する。

【0026】ステップ2において、処理装置100のシステム間排他手段202は、指定された識別子で排他確保をする。即ち、システム間排他手段202は、「識別子」＝“000001”を排他状態にする。

【0027】次に、ステップ3において、処理装置100の更新状態確認手段203は、指定された識別子は他のシステム系で更新済みか否かを判断する。即ち、更新

けられている共用装置110上のグローバルメモリ212が他システムからアクセスされているかどうかチェックする。更新済みである場合には、ステップ4に進み、更新済みでない場合には、ステップ13に進む。ステップ4、5については、後述する。

【0028】ここでは、処理装置100からの「識別子」＝“000001”に対するアクセスが初めてのものであるとして、ステップ13に進む。

【0029】ステップ13において、共用装置110の識別子割当て手段210は、「識別子」＝“000001”に対して、処理装置100(SYS A)から割当て要求がきたことを記録し、共有メモリ管理手段211は、「識別子」＝“000001”に対応する共用装置110上のグローバルメモリ212を長さ＝“1000”X分確保する。

【0030】次に、ステップ6において、処理装置100は、処理装置100上のロック要求処理では、初めてのアクセスであり、「更新ベクトル」上では他系による更新がない状況であるため、データを転送せず、ロック要求を完了する。

【0031】以上のロック要求の処理により、処理装置100上のプログラムは、この識別子と対応付けられた領域のアクセスが可能となる。プログラムは、この領域を特別な命令を使用せずに、自システム内の領域をアクセスするのと同様な手順で容易に、参照・更新できる。

【0032】本実施形態によれば、特別な命令を使用せずに、自システム内の領域をアクセスするのと同様な手順で容易に、参照・更新できる。

【0033】次に、図3を用いて、複数システム間で定義したシステム間の共通メモリ領域をアンロック要求するときの手順について説明する。図3は、本発明の一実施形態によるデータ共用方式におけるアンロック要求処理の手順を示すフローチャートである。

【0034】図3のステップ7において、処理装置100上のプログラムは、共通領域のアクセスを完了すると、「識別子」＝“000001”を入力として、アンロック要求を発行する。

【0035】ステップ8において、処理装置100は、更新指示があったか否かを判断する。更新指示があれば、ステップ9に進み、更新指示がなければ、ステップ11にジャンプする。処理装置100上のプログラムが、このメモリ領域を参照のみした場合には、更新指示はなされない。

【0036】ここで、処理装置100上のプログラムからの「識別子」＝“000001”へのアクセスは初めてである場合には、グローバルメモリ212上にデータを作成していないため、処理装置100は、更新指示を行う。

【0037】次に、ステップ9において、識別子に対応

送する。即ち、データ転送手段204は、識別子="00001"に対応するローカルメモリ206の「アドレス」="10002000" Xから「アドレス」="10002FFF" Xの内容を、グローバルメモリ212上の「アドレス」="00000000" Xから「アドレス」="00001000" Xへ転送する。

【0038】次に、ステップ10において、更新ベクトルをセットする。即ち、更新ベクトル設定手段205は、自系からの更新が完了したことを他系が認識できるように、共有メモリ管理手段211の管理する更新ベクトルのビットを"0" Bから"1" Bに設定する。ここで、共有メモリ管理手段211の「識別子」="000001"に対応する「更新ベクトル」を"000"から"100"に設定する。処理装置100である「SYS A」は、識別子割当て手段210の「割当てシステム名称」の欄の記録からして一番最初に割り当てられているため、更新ベクトルの最上位ビットを"0" Bから"1" Bに設定する。

【0039】次に、ステップ11において、システム間開放手段202は、指定された識別子による排他解放を行う。即ち、「識別子」="000001"は排他開放される。

【0040】ステップ12において、処理装置100上のプログラムは、アンロック要求を完了する。

【0041】なお、ステップ8において、更新指示がなかった場合には、ローカルメモリ206上のデータは、グローバルメモリ212上に転送されず、ステップ11において、システム間排他手段202は、指定された識別子によるシステム間の排他解放を実行し、ステップ12において、プログラムはアンロック要求処理を終了する。

【0042】次に、上述したようにして、処理装置100によって割り当てられた共有メモリを、他系、例えば、処理装置120が参照・更新する場合のロック要求処理について、同じく、図2を用いて説明する。

【0043】図2のステップ1において、処理装置120上のプログラムは、予め確保してあるアドレス空間上のローカルメモリ226の「アドレス」="0E001000" Xから「長さ」="1000" X分の領域を、システム間の共通メモリとしてアクセスするために、処理装置100上のプログラムとあらかじめ決めておいた「識別子」="000001"によりロック要求を発行する。

【0044】ステップ2において、処理装置120のシステム間排他手段222は、指定された識別子で排他確保をする。即ち、システム間排他手段222は、「識別子」="000001"を排他状態にする。

【0045】次に、ステップ3において、処理装置120の更新状態確認手段223は、指定された識別子は他

更新状態確認手段223が、共有メモリ管理手段211の管理する更新ベクトルの内容をチェックすることにより、「識別子」="000001"は、既に、処理装置100からアクセスされているので、ステップ4に進む。

【0046】ステップ4において、データ転送手段224は、識別子に対応するグローバルメモリ212からローカルメモリ226へデータを転送する。即ち、データ転送手段224は、「識別子」="000001"に対応するグローバルメモリ212の「アドレス」="00000000" X~"00000FFF" Xの内容を、処理装置120上のローカルメモリ226の「アドレス」="0E001000" X~"0E001FFF" Xに転送する。このデータ転送処理により、処理装置100上の「識別子」="000001"と処理装置120上の「識別子」="000001"上のデータが一致する。

【0047】ステップ5において、処理装置120の更新ベクトル設定手段225は、更新ベクトルをリセットする。即ち、更新ベクトル設定手段225は、処理装置120上ではデータ転送後、このデータを再度読み込まないように、「更新ベクトル」を"1" Bから"0" Bへリセットする。その結果、「識別子」="000001"に対する「更新ベクトル」は、"100"から"000"にリセットされる。

【0048】ステップ6において、ロック要求を完了し、要求プログラム元に戻りターンする。そして、ロック要求が完了した処理装置120上のプログラムは、「識別子」="000001"と対応付けられたローカルメモリ226上のデータに対し、参照・更新を実行する。

【0049】次に、図3に示したアンロック要求処理を実行する。ステップ7において、アンロック要求を発行し、ステップ8において、更新指示があるか否かを判断する。参照・更新実行が、参照のみの場合は、更新指示を行わずに、ステップ11において、排他解放し、ステップ12において、アンロック処理を完了する。

【0050】また、ステップ8において、更新指示があると判断された場合には、ステップ9において、ローカルメモリからグローバルメモリへデータを転送し、ステップ10において、更新ベクトルリセットした後、ステップ11において、排他解放し、ステップ12において、アンロック処理を完了する。

【0051】以上説明したような処理手順により、ロック→参照・更新→アンロックを繰り返すことにより、システム間で共通な領域をアクセスすることができる。ロック→参照・更新→アンロックの手順は、複数の仮想アドレス空間を持つシステムにおいて、全空間から共通の領域を参照・更新する場合の手順と同じである。システム間の共通メモリも、従来の共通領域と同等の手軽さで

【0052】以上説明したように、本実施形態によれば、プログラムは、特別な命令を使用せずに、自システム内の領域をアクセスすると同様な手順で容易に、共通メモリの領域を参照・更新できるようになる。

【0053】次に、図4を用いて、排他の最小単位である識別子を複数まとめてグループ化してアクセスする方式について説明する。図4は、本発明の他の実施形態によるデータ共用方式を実現する複合コンピュータシステムの構成図である。なお、図1と同一符号は、同一部分を表しており、その構成及び機能は同一であるため、詳細な説明は省略する。

【0054】なお、本実施形態では、「識別子」＝"000001"，"000002"を一つのグループとしてまとめてアクセスし、「識別子」＝"000003"，"000004"を一つのグループとしてまとめてアクセスするようにしている。

【0055】処理装置100，120は、それぞれ、「識別子」＝"000001"，"000002"は、一つのグループとしてまとめられているものとして認識しており、処理装置100，120上のプログラム間では、あらかじめ、「識別子」＝"000001"，"000002"のグループをアクセスする場合は、代表の「識別子」として"000001"を用い、「識別子」＝"000003"，"000004"のグループをアクセスする場合は、代表の「識別子」として"000003"を用いるようにしている。

【0056】ここで、ローカル領域割当て手段201の中には、例えば、ローカルメモリ206のアドレス"10002000" Xからアドレス"10002FFF" Xまでの長さ"1000" Xの領域を「識別子」＝"000001"で管理し、ローカルメモリ206のアドレス"10003000" Xからアドレス"10003FFF" Xまでの長さ"1000" Xの領域を「識別子」＝"000002"で管理し、ローカルメモリ206のアドレス"10004000" Xからアドレス"10004FFF" Xまでの長さ"1000" Xの領域を「識別子」＝"000003"で管理し、ローカルメモリ206のアドレス"10005000" Xからアドレス"10005FFF" Xまでの長さ"1000" Xの領域を「識別子」＝"000004"で管理しているものとする。「識別子」＝"000001"，…，"000004"で管理されるメモリ領域は、ローカルメモリ206に図示した領域である。

【0057】また、ローカル領域割当て手段221の中には、例えば、ローカルメモリ226のアドレス"0E003000" Xからアドレス"0E003FFF" Xまでの長さ"1000" Xの領域を「識別子」＝"000001"で管理し、ローカルメモリ226のアドレス"0E004000" Xからアドレス"0E004F

＝"000002"で管理し、ローカルメモリ226のアドレス"0E001000" Xからアドレス"0E001FFF" Xまでの長さ"1000" Xの領域を「識別子」＝"000003"で管理し、ローカルメモリ226のアドレス"0E002000" Xからアドレス"0E002FFF" Xまでの長さ"1000" Xの領域を「識別子」＝"000004"で管理しているものとする。「識別子」＝"000001"，…，"000004"で管理されるメモリ領域は、ローカルメモリ206に図示した領域である。

【0058】さらに、識別子割当て手段210の中には、例えば、「識別子」＝"000001"は、処理装置100である"SYS A"，処理装置120である"SYS B"に割り当てられており、「識別子」＝"000002"は、"SYS A"及び"SYS B"に割り当てられており、「識別子」＝"000003"は、"SYS B"及び"SYS A"に割り当てられており、「識別子」＝"000004"は、"SYS B"及び"SYS A"に割り当てられていることが記録されている。

【0059】また、共有メモリ管理手段211の中には、例えば、グローバルメモリ212のアドレス"00000000" Xからアドレス"00000FFF" Xまでの領域を「識別子」＝"000001"で管理し、「更新ベクトル」が"000"であることが記録され、グローバルメモリ212のアドレス"00006500" Xからアドレス"000074FF" Xまでの領域を「識別子」＝"000002"で管理し、「更新ベクトル」が"010"であることが記録され、グローバルメモリ212のアドレス"00004000" Xからアドレス"00004FFF" Xまでの領域を「識別子」＝"000003"で管理し、「更新ベクトル」が"000"であることが記録され、グローバルメモリ212のアドレス"00008000" Xからアドレス"00008FFF" Xまでの領域を「識別子」＝"000001"で管理し、「更新ベクトル」が"100"であることが記録されている。

【0060】複数の識別子をグループ化してロック要求する場合の処理について、再び、図2を用いて説明する。

【0061】図2のステップ1において、処理装置100上のプログラムは、予め確保してあるアドレス空間上のローカルメモリ206の「アドレス」＝"10004000" Xから「長さ」＝"1000" X分の領域及びローカルメモリ206の「アドレス」＝"10005000" Xから「長さ」＝"1000" X分の領域を、システム間の共通メモリとしてアクセスするために、処理装置120上のプログラムとあらかじめ決めておいた「識別子」＝"000003"によりロック要求を発行

別子」＝"000004"で管理される領域をロックする訳であるが、ここでは、代表となる識別子でロック要求するようにしている。

【0062】ステップ2において、処理装置100のシステム間排他手段202は、指定された識別子で排他確保をする。即ち、システム間排他手段202は、「識別子」＝"000003"を排他状態にする。その結果として、「識別子」＝"000004"が排他状態とはなっていないことになるが、処理装置100、120のいづれもが、「識別子」＝"000003"と「識別子」＝"000004"をグループ化して扱うように取り決めてあるので、「識別子」＝"000003"が排他状態にあるときには、「識別子」＝"000004"に対して排他要求をすることはなく、「識別子」＝"000004"については、実質的に排他が確保されている。

【0063】次に、ステップ3において、処理装置100の更新状態確認手段203は、指定された識別子は他のシステム系で更新済みか否かを判断する。即ち、更新状態確認手段203は、この指定された識別子で対応付けられている共用装置110上のグローバルメモリ212が他システムからアクセスされているかどうかチェックする。

【0064】ここで、処理装置100から、識別子＝"000003"でロック要求がきた場合でも、更新状態確認手段203は、代表の識別子ではなく、「識別子」＝"000003"、"000004"の両方に対してチェックを行い更新状態を確認する。図4に示す状態では、共有メモリ管理手段211の中の識別子＝"000003"に対する更新ベクトルは、"000"であり、更新されていないが、識別子＝"000004"に対する更新ベクトルは、"100"であり、処理装置120(SYS B)から更新されていることがわかるため、ステップ4に進む。

【0065】ステップ4において、データ転送手段204は、グローバルメモリ212からローカルメモリ206に対してデータを転送する。

【0066】次に、ステップ5において、更新ベクトル設定手段205は、共有メモリ管理手段211の中の識別子＝"000004"に対する更新ベクトルをリセットする。

【0067】次に、ステップ6において、処理装置100は、ロック要求を完了し、読みだしたデータに対して、参照・更新を行う。

【0068】以上のロック要求の処理により、処理装置100上のプログラムは、この識別子と対応付けられた複数の領域のアクセスが可能となる。参照・更新される単位(領域)を複数設けたとしても、排他オーバーヘッドは一回で済むため、排他オーバーヘッドを削減できる。

【0069】プログラムは、この領域を特別な命令を使

な手順で容易に、参照・更新できる。

【0070】アンロック要求に対する処理は、図3を用いて前述したとおりであり、排他確保されているのは、代表となる識別子についてであるので、ステップ11における排他開放も、排他確保された識別子についてのみ排他開放される。

【0071】本実施形態によれば、特別な命令を使用せずに、自システム内の領域をアクセスするのと同様な手順で容易に、参照・更新できる。

【0072】また、複数の領域に対して排他確保する際の排他オーバーヘッドを削減できる。

【0073】次に、図5を用いて、処理装置上のプログラムがシステム間の共通メモリを参照・更新する場合に、共用装置へのアクセスと、ローカルメモリのアクセスを非同期にする方法について説明する。図5は、本発明のその他の実施形態によるデータ共用方式を実現する複合コンピュータシステムの構成図である。なお、図1と同一符号は、同一部分を表しており、その構成及び機能な同一であるため、詳細な説明は省略する。

【0074】処理装置100上では、プログラムA207は、システム間の共通メモリと対応付けられたローカルメモリ206を参照・更新しながら動作するプログラムとして稼働している。

【0075】ここで、処理装置100の中のローカル領域割当て手段が管理するローカルメモリ206の領域の割当ては、図4に示したものと同様とする。

【0076】また、処理装置120上では、プログラムB227は、システム間の共通メモリと対応付けられたローカルメモリ226を参照・更新しながら動作するプログラムとして稼働している。

【0077】そして、処理装置120の中のローカル領域割当て手段が管理するローカルメモリ226の領域の割当ては、図4に示したものと同様とする。

【0078】さらに、識別子割当て手段210における割当てシステムの名称は、図4に示したものと同様とする。

【0079】また、共有メモリ管理手段211における共有メモリであるグローバルメモリ212の管理状況も図4に示したものと同様とする。

【0080】プログラムA207とプログラムB227が共有するデータは、ユーザデータのように全システム間で完全性が保証されていなければならないデータではなく、例えば、処理装置100、120上のCPUのビジー率や、共用資源の稼働情報など、多少の誤差が許容されるデータである。

【0081】そのようなデータを共有する場合は、データのやりとりが定期的に発生するため、共用装置110へのアクセスオーバーヘッドまたは排他オーバーヘッドは極力減らすことが望ましい。そこで、共用装置110への

動作することなく、全く別のプログラムにより比較的オーバーヘッドの少ない周期で定期的に共用装置110へのデータアクセスを行い、ローカルメモリ206、226上へデータを反映するようにしている。

【0082】プログラムA207及びプログラムB227は、ローカルメモリ206、226上へのデータの反映契機とは別の契機で動作し、必要なときにローカルメモリ206、226だけ参照・更新する。

【0083】このようにして、ローカルメモリへのアクセスと、グローバルメモリへのアクセス契機を分けることにより、共用装置へのオーバーヘッドを減らすことが可能となる。

【0084】本実施形態によれば、特別な命令を使用せずに、自システム内の領域をアクセスするのと同様な手順で容易に、参照・更新できる。

【0085】また、各コンピュータシステム上でデータを参照・更新する契機と、共用装置上のデータをコンピュータシステムに反映する契機をわけることにより、共用装置へのオーバーヘッドを低減できる。

【0086】

【発明の効果】本発明によれば、複数のコンピュータシステム間でメモリを共有する場合、従来のシステム内の共通領域へのアクセスと同等の手順により、参照・更新が可能となる。

【図面の簡単な説明】

【図1】本発明の一実施形態によるデータ共用方式を実

現する複合コンピュータシステムの構成図である。

【図2】本発明の一実施形態によるデータ共用方式におけるロック要求処理の手順を示すフローチャートである。

【図3】本発明の一実施形態によるデータ共用方式におけるアンロック要求処理の手順を示すフローチャートである。

【図4】本発明の他の実施形態によるデータ共用方式を実現する複合コンピュータシステムの構成図である。

【図5】本発明のその他の実施形態によるデータ共用方式を実現する複合コンピュータシステムの構成図である。

【符号の説明】

100、120…処理装置

110…共用装置

201、221…ローカル領域割当て手段

202、222…システム間排他手段

203、223…更新状態確認手段

204、224…データ転送手段

205、225…更新ベクトル設定手段

206、226…ローカルメモリ

207…プログラムA

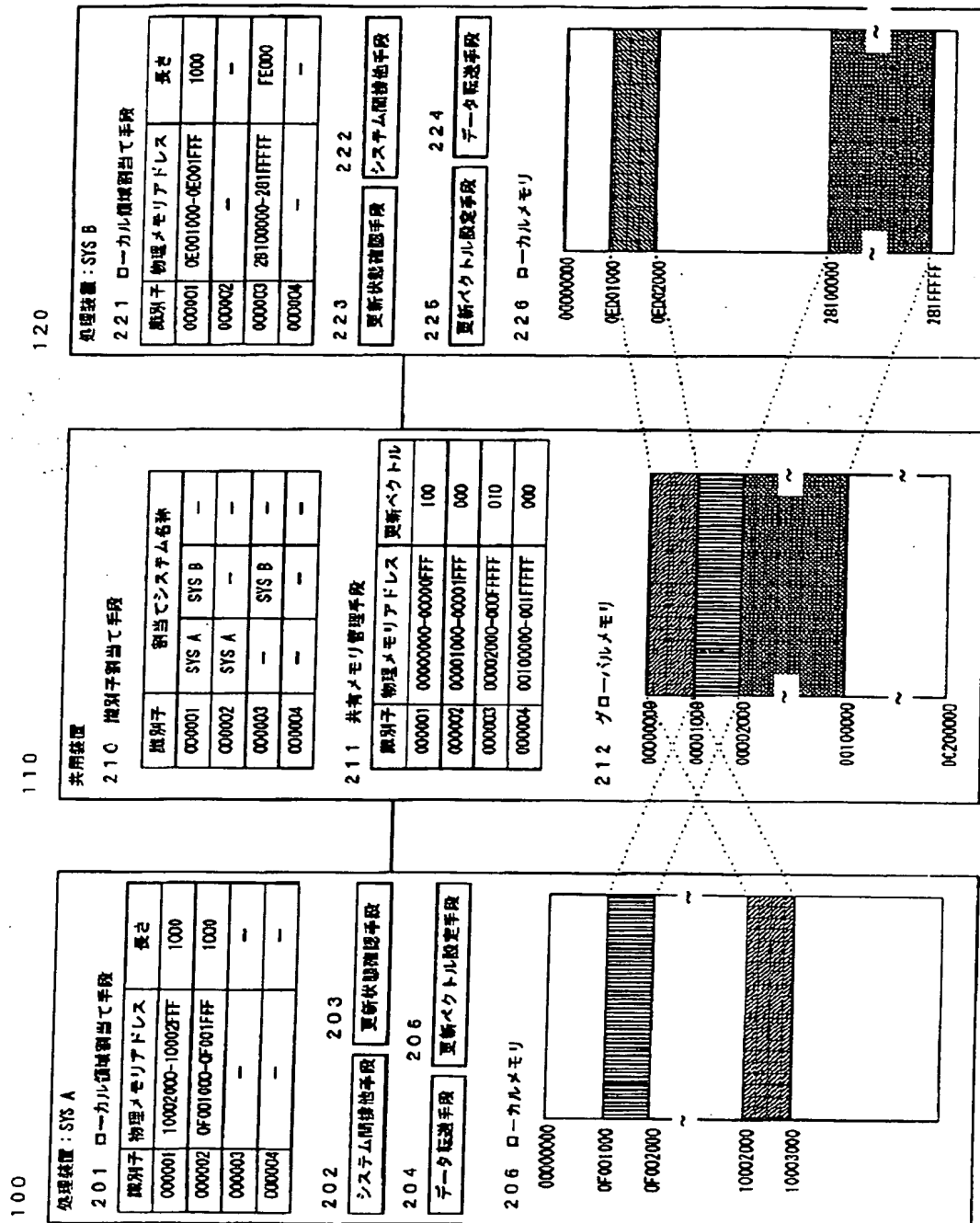
227…プログラムB

210…識別子割当て手段

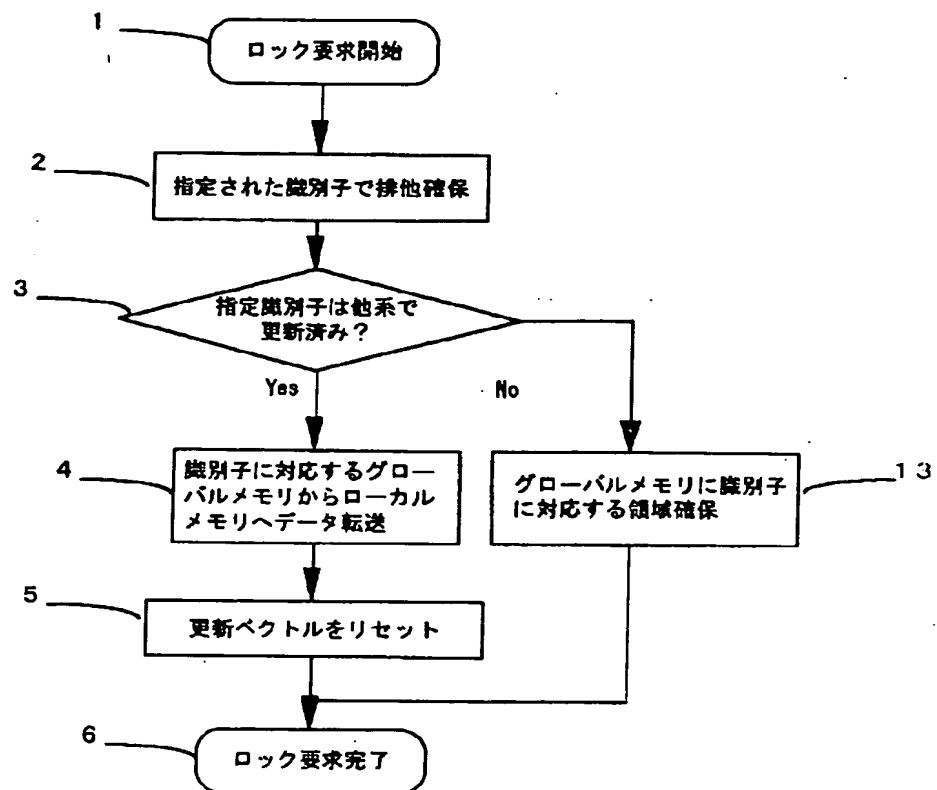
211…共用メモリ管理手段

212…グローバルメモリ

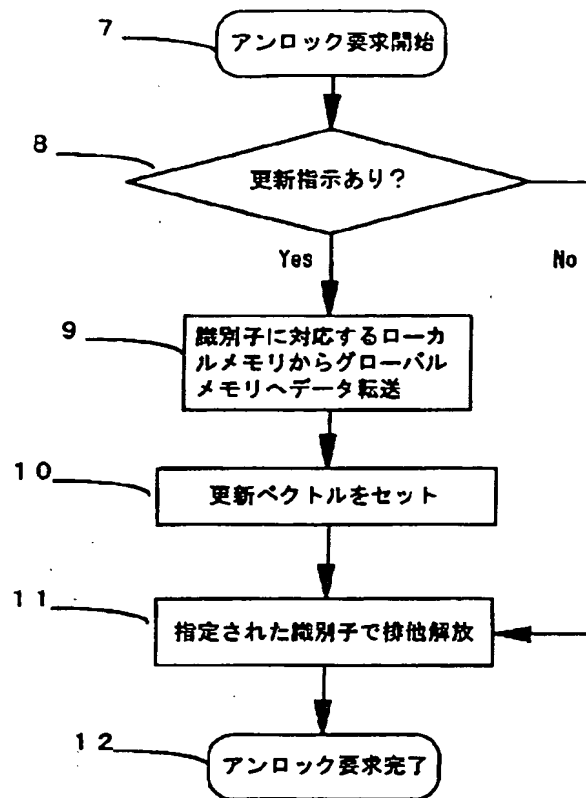
【図1】



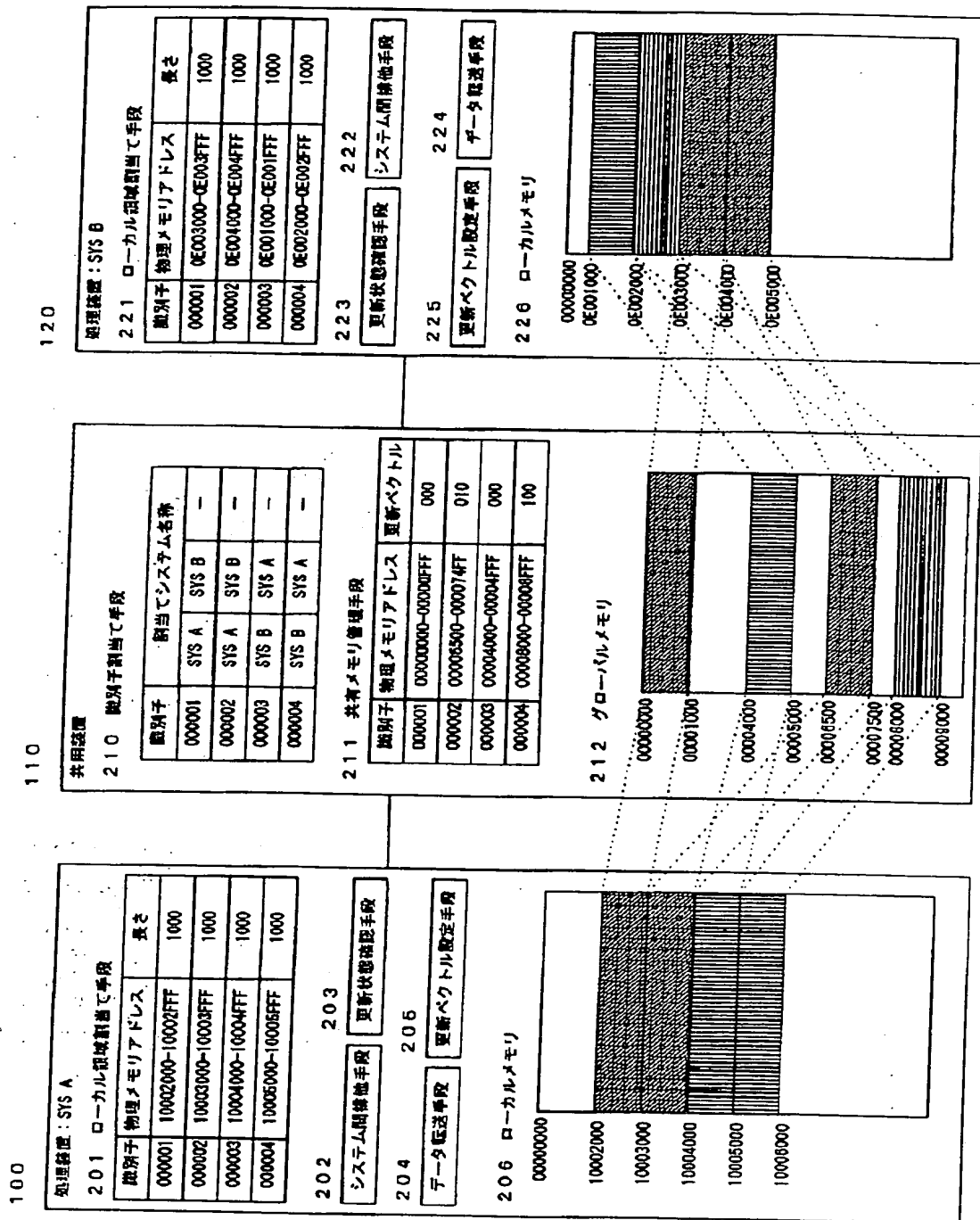
【図2】



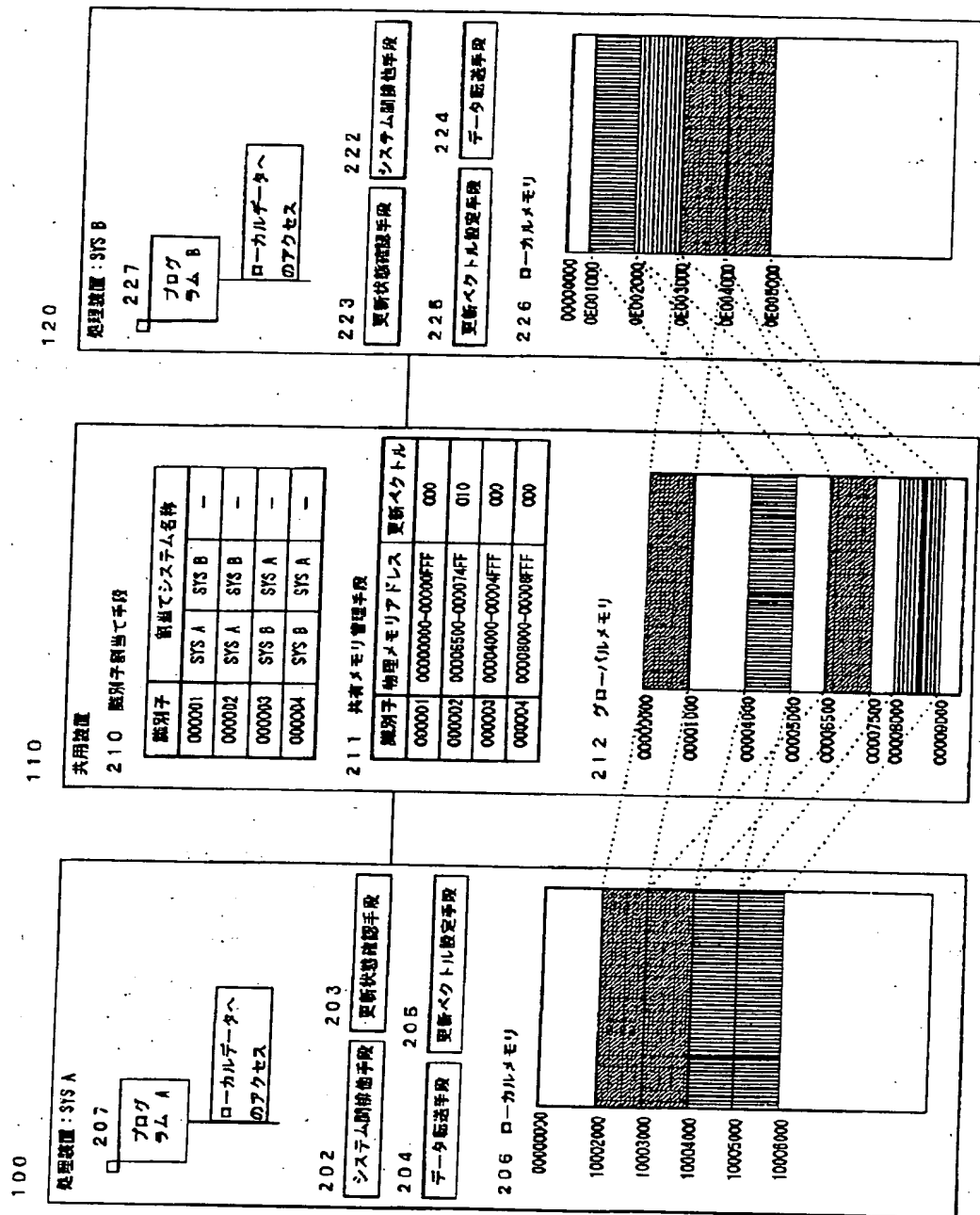
【図3】



【図4】



【図5】



フロントページの続き

(72)発明者 田巻 貴久子

神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作所ソフトウェア開発本部内



US005978839A

United States Patent [19]

Okuhara et al.

[11] Patent Number: 5,978,839

[45] Date of Patent: Nov. 2, 1999

[54] **DATA SHARING METHOD IN A PLURALITY OF COMPUTER SYSTEMS**

[75] Inventors: Susumu Okuhara; Hiroshi Morishima; Shingo Maeda; Kikuko Morishima, all of Yokohama, Japan

[73] Assignee: Hitachi, Ltd., Tokyo, Japan

[21] Appl. No.: 08/859,400

[22] Filed: May 20, 1997

[30] **Foreign Application Priority Data**

May 21, 1996 [JP] Japan 8-126083

[51] Int. Cl.⁶ G06F 13/00; G06F 15/167

[52] U.S. Cl. 709/215; 711/153; 711/202

[58] Field of Search 395/200.45, 200.43, 395/200.44, 200.46, 726, 427; 711/152, 153, 162, 163, 147, 148, 150, 202; 710/200; 709/215, 213, 214, 216

[56] **References Cited****U.S. PATENT DOCUMENTS**

5,485,594 1/1996 Foster 395/427
5,522,045 5/1996 Sandberg 709/215
5,829,041 10/1998 Okamoto et al. 711/147

OTHER PUBLICATIONS

Program Product VOS3/AS Center Operation—JSS3—, pp. 226–239, Hitachi Computers General Information/User Guide, Dec. 1995.

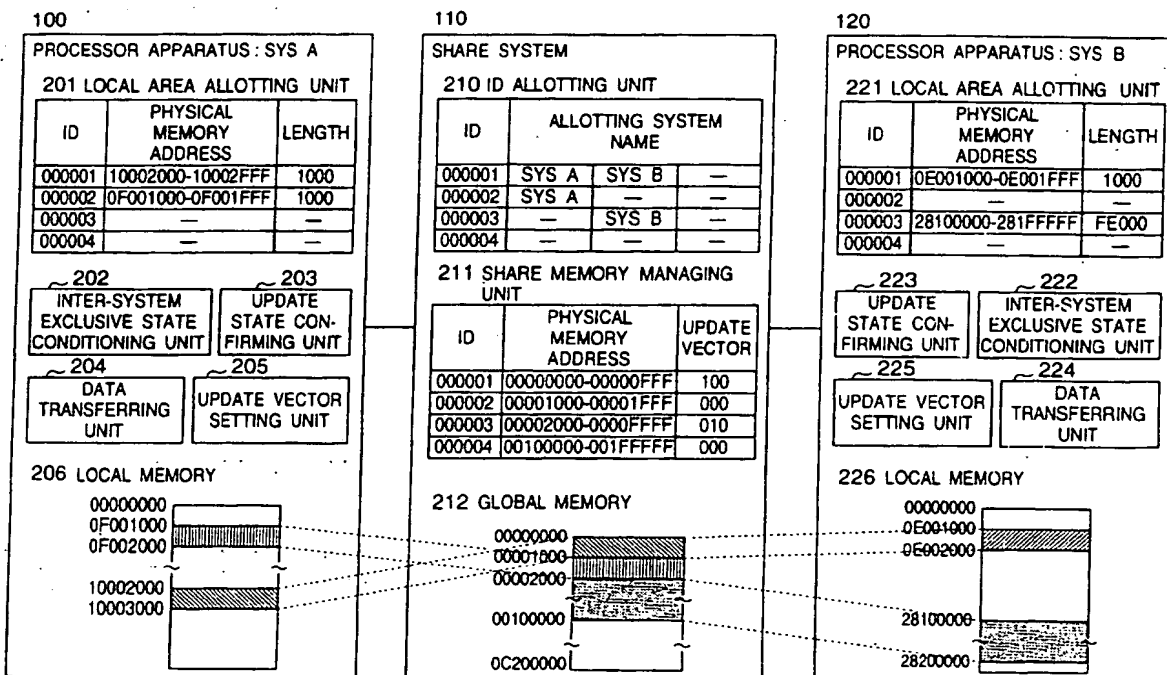
Primary Examiner—Dung C. Dinh

Attorney, Agent, or Firm—Antonelli, Terry, Stout & Kraus, LLP

[57] **ABSTRACT**

Processor apparatuses share data, used in common for a plurality of systems, in a global memory in a share system. The global memory is divided into constant units and the individual division units are managed by using identifiers which are definitely determined as viewed from the processor apparatuses and the share system. When an inter-system exclusive state conditioning unit of a processor apparatus designates a desired area on an address space in the processor apparatus by using an identifier to obtain a locked resource, an update state confirming unit checks whether data in a corresponding unit in the share system is updated, and when the data is updated, a data transferring unit transfers the data in the corresponding unit in the share system to the address space. Accordingly, a program operating on each computer system can easily access the data shared by the systems without being aware of the presence of the share system. When the locked resource is released, a data transferring unit transfers the data to the corresponding unit in the share system if the data in the address space is updated.

6 Claims, 5 Drawing Sheets



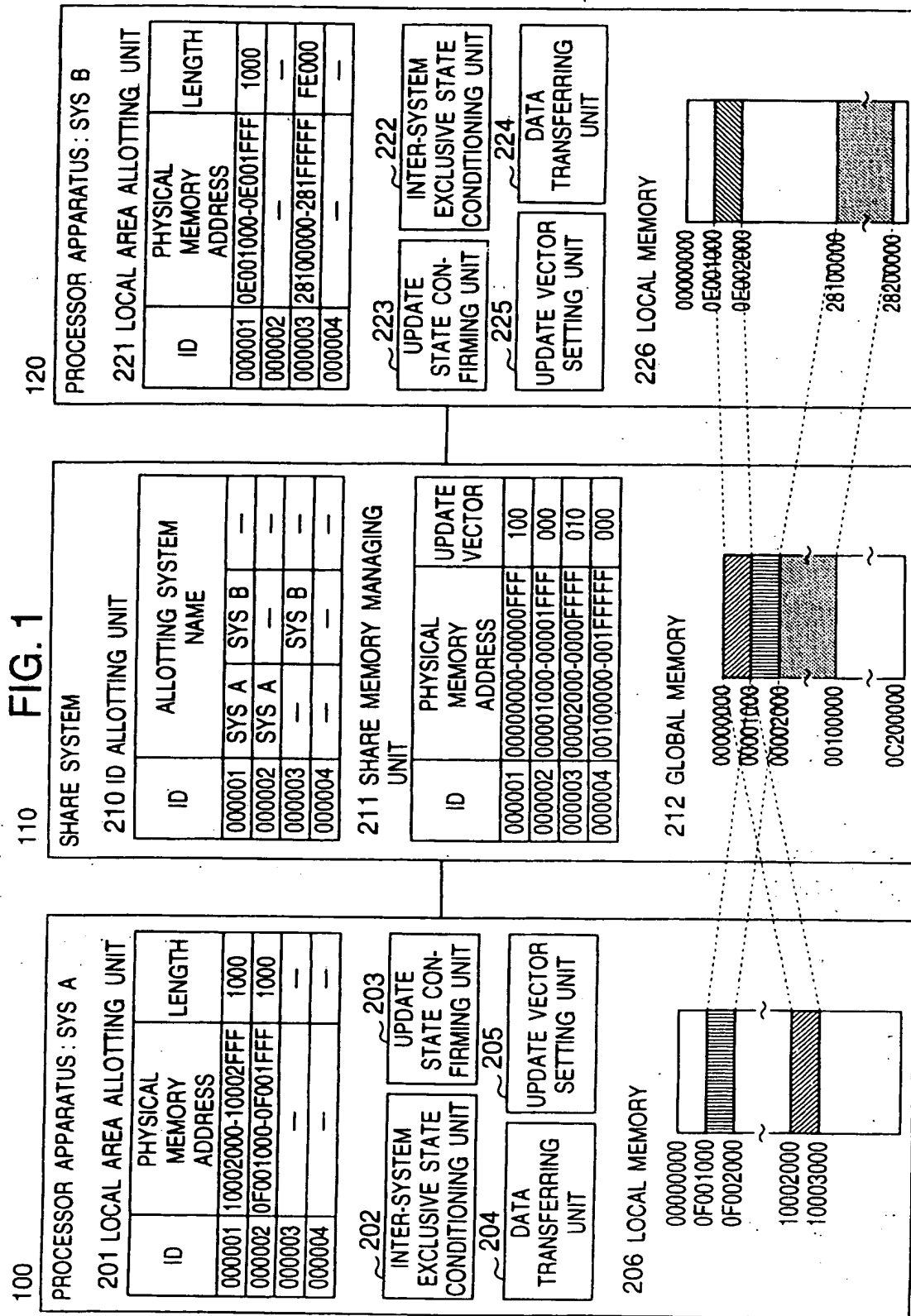


FIG. 2

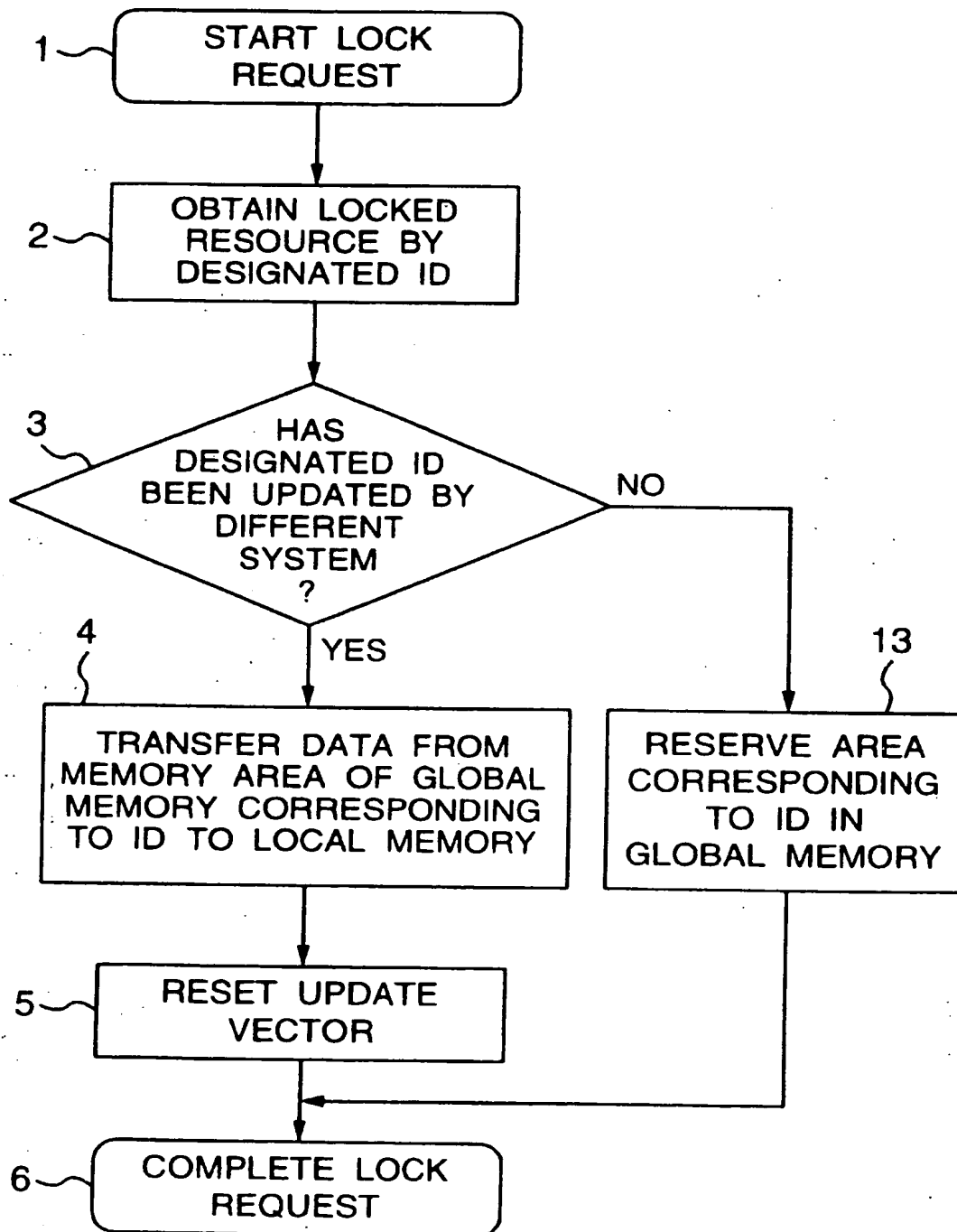
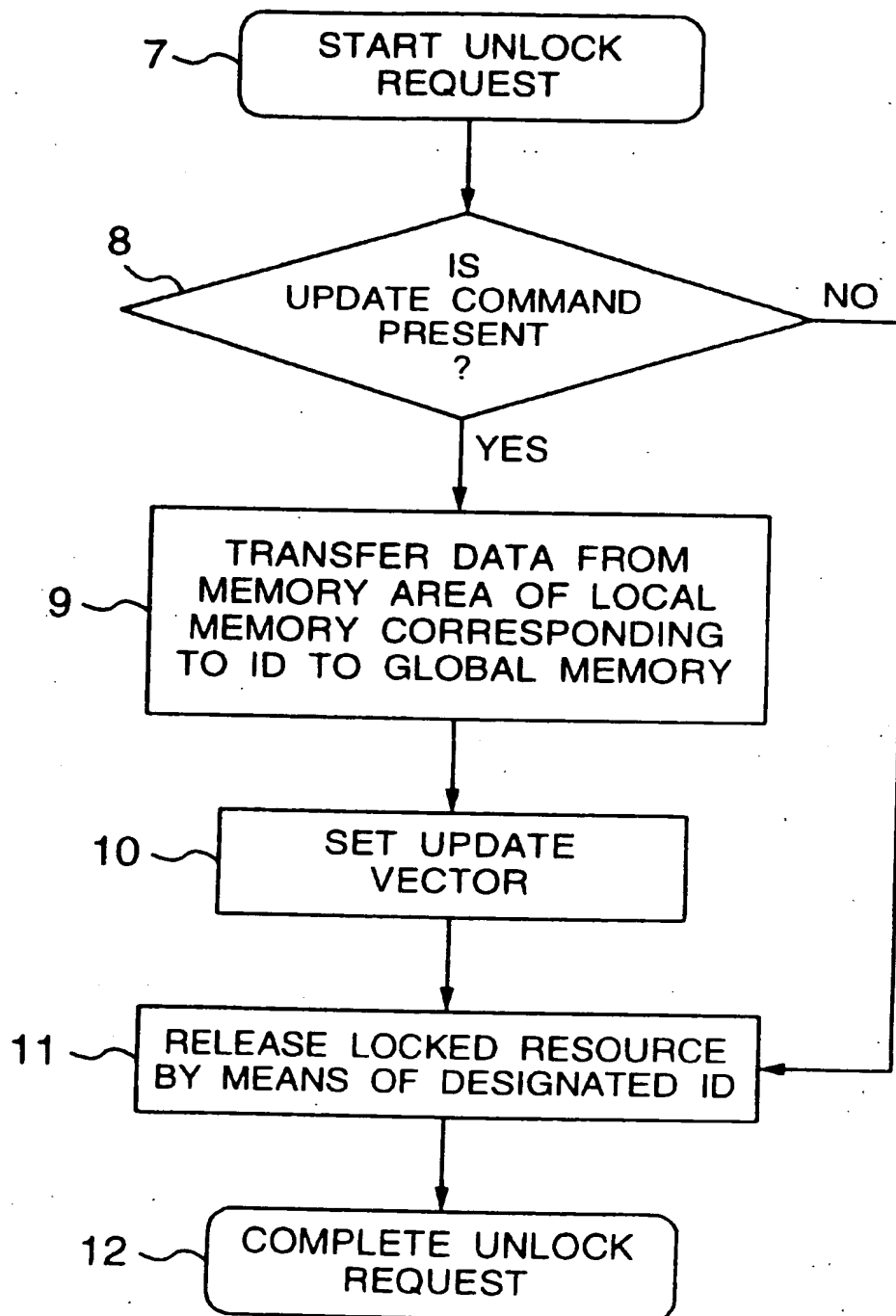
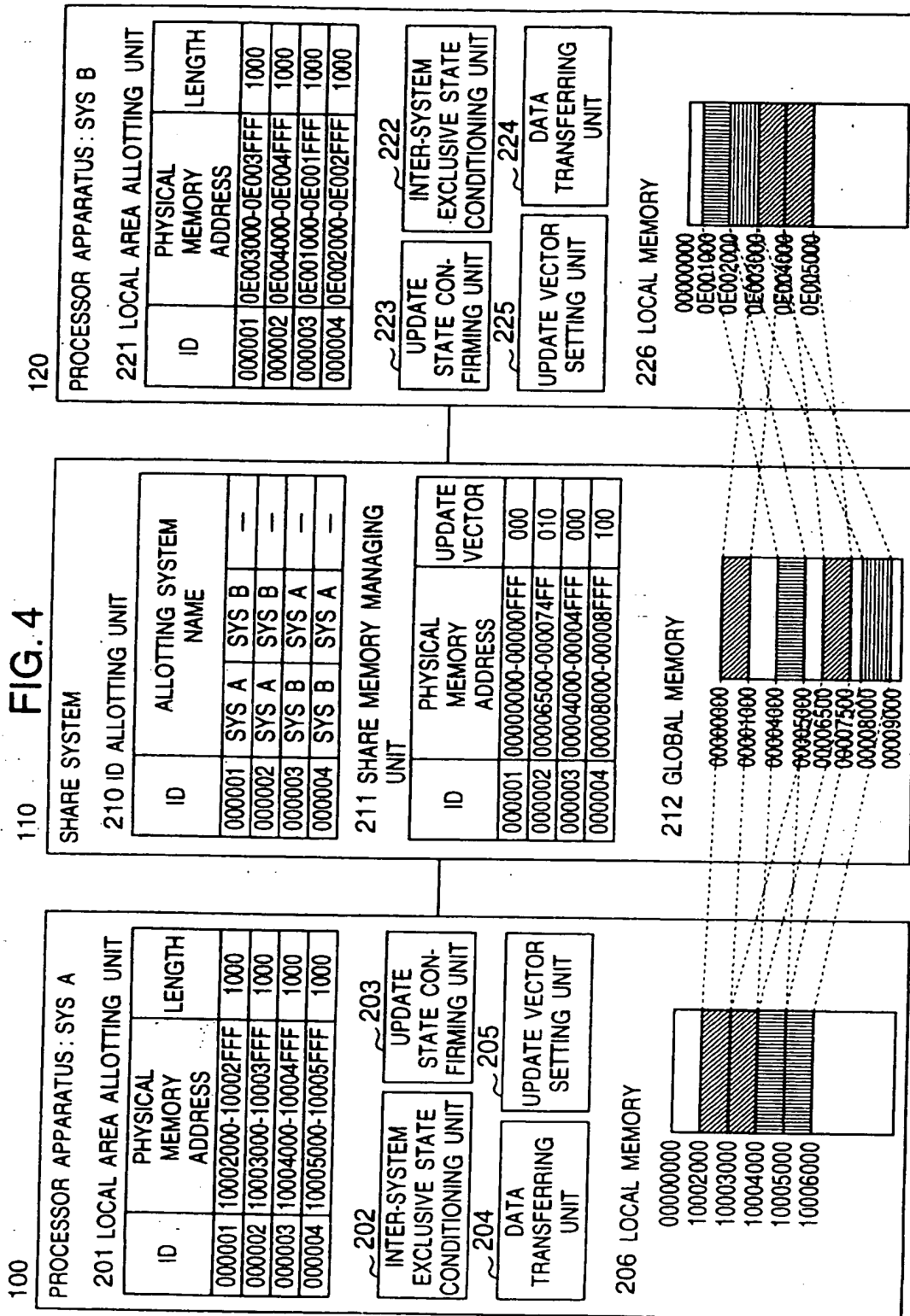
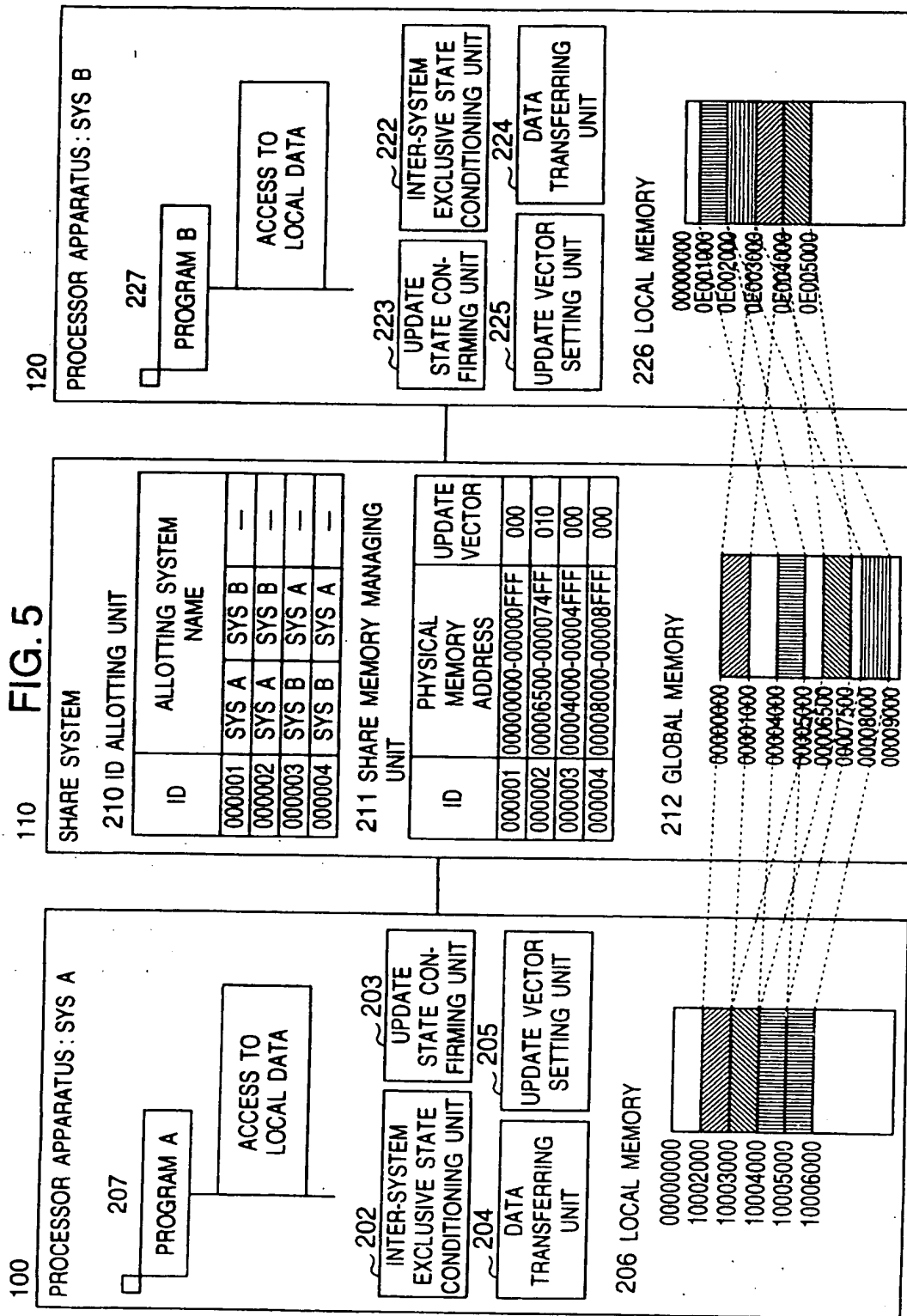


FIG. 3







DATA SHARING METHOD IN A PLURALITY OF COMPUTER SYSTEMS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to data sharing methods and more particularly to a data sharing method in which data used in common for a plurality of computer system is shared by a share system.

2. Description of the Related Art

Conventionally, when data on an address space is shared by a plurality of computer system, a share input/output system is occupied, data is once written to a predetermined position and thereafter the occupied state is released. Subsequently, when another system utilizes the data, the input/output system is similarly occupied, the data at the predetermined position is read and transferred to an area precedently reserved on an address space and thereafter, the occupied state is release. In other words, the individual computer system once occupy the input/output system to perform read/write of data and thereafter release the occupation. Used as the input/output system shared by the plurality of system is, for example, a magnetic storage system, a semiconductor storage system or a stand-alone extended storage system.

A method of sharing data by a plurality of system by using the stand-alone extended storage is described in, for example, a manual "Program Product VOS3/AS Center Operation-JSS3-" published by Hitachi, Ltd., December, 1995, pp. 226-239.

However, when the input/output system serving in a share system is occupied by a conventional system, the effect of overhead due to the exclusive state of the input/output system is increased, raising a problem that even when data of a relatively small amount is accessed frequently, the overhead due to the exclusive state and the access overhead are cumulated and cannot be neglected.

SUMMARY OF THE INVENTION

An object of the present invention is to provide a data sharing method in which a program operating on each computer system can easily access data shared by systems through the procedure similar to that for accessing an area in a system of its own without being aware of the presence of a share system.

To accomplish the above object, the present invention discloses a data sharing method in which data used in common for a plurality of computer systems is shared by a share memory in the share system.

More specifically, the method has a first step of dividing the share memory into constant units and managing the individual division units by using identifiers which are definitely determined as viewed from the plurality of computer systems and the share system. The method has a second step of checking, when a desired area on an address space in a computer system is designated by using an identifier to obtain a locked resource, whether data in a corresponding unit in the share memory is updated and a third step of transferring data in the corresponding unit in the share memory to the address space when the data is updated, whereby when the data in the address space is updated at the time that the locked resource is released, the data which has been transferred to the address space is transferred to the corresponding unit in the share memory. According to the data sharing method of the present invention, a program

operating on each computer system can easily access data shared by the systems through the procedure similar to that for accessing an area in a system of its own without being aware of the presence of the shared apparatus.

Preferably, in the data sharing method of the present invention, the computer system may access a group of a plurality of identifiers and obtain a locked resource in the share system by using a representative identifier of the identifier group. In this case, the overhead due to the exclusive state can be reduced.

Preferably, in the data sharing method of the present invention, the access timing at which the computer system looks-up/updates data on the computer system may be made to be asynchronous to the access timing at which data on the share system is reflected upon the computer system. In this case, the overhead for the share system can further be reduced.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic diagram of a composite computer system realizing a data sharing method according to an embodiment of the present invention.

FIG. 2 is a flow chart showing the procedure of the lock requesting process in the data sharing method according to the embodiment of the present invention.

FIG. 3 is a flow chart showing the procedure of the unlock requesting process in the data sharing method according to the embodiment of the present invention.

FIG. 4 is a schematic diagram of a composite computer system realizing a data sharing method according to another embodiment of the present invention.

FIG. 5 is a schematic diagram of a composite computer system realizing a data sharing method according to still another embodiment of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

A data sharing method according to an embodiment of the present invention will now be described with reference to FIGS. 1 to 3.

FIG. 1 schematically shows a composite computer system realizing the data sharing method according to the embodiment of the present invention.

A processor apparatus 100 has a local memory 206 which can be looked up using an address space on a system comprised of the processor apparatus 100. A share system 110 which can be shared for use by other processor apparatus or system than the processor apparatus 100 has a global memory 212 usable by the processor apparatus 100. The processor apparatus 100 also has a local area allotting unit 201 for managing memory areas which correspond, in 1:1 relationship, to specified memory areas of global memory 212 managed by different identifiers on the share system 110 and other memory areas on an address space of the local memory 206 in its own system. The local area allotting unit 201 manages the memory areas in the local memory 206 by using a table format having items "identifier (ID)", "physical memory address" and "length".

In the local area allotting unit 201, for example, an area of a length of '1000'X extending from an address '10002000'X ("X" being an abbreviation for the hexadecimal number) to an address '10002FFF'X of the local memory 206 is managed by an ID='000001' and an area of a length of '1000'X extending from an address '0F001000'X to an address '0F001FFF'X of the local memory 206 is managed by an ID='000002'.

The memory areas managed by the ID's '000001' and '000002' are those depicted in the local memory 206.

Further, the processor apparatus 100 has an inter-system exclusive state conditioning unit 202 for conditioning an exclusive state to a different system to obtain/release a locked resource during transfer of data to the share system 110, an update state confirming unit 203 for checking whether data on the share system is updated by the different system, a data transferring unit 204 for transferring data from the local memory 206 on the processor apparatus 100 to the global memory 212 on the share system 110 or from the global memory 212 on the share system 110 to the local memory 206 on the processor apparatus 100, and an update vector setting unit 205 for setting an update vector used to record the update state of the different system on the share system 110. The update vector will be described later.

A processor apparatus 120 is constructed similarly to the processor apparatus 100, having a local area allotting unit 221, an inter-system exclusive state conditioning unit 222, an update state confirming unit 223, a data transferring unit 224, an update vector setting unit 225 and a local memory 226. Each unit has the same function as that of each corresponding unit in the processor apparatus 100 and will not be detailed.

It is recorded in the local area allotting unit 221 that, for example, an area of a length of '1000'X extending from an address '0E001000'X to an address '0E001FFF'X of the local memory 226 is managed by an ID='000001' and an area of a length of 'FE000'X extending from an address '28100000'X to an address '281FFFFF'X of the local memory 226 is managed by an ID='000003'.

The memory areas managed by the ID's '000001' and '000003' are those depicted in the local memory 226.

The share system 110 is coupled to the processor apparatus (SYS A) 100 and the processor apparatus (SYS B) 120 and incorporates its internal memory to fulfil the function of operating data on the memory in accordance with commands from the coupled processor apparatuses 100 and 120. For example, the share system is constructed of a magnetic storage system, a semi-conductor storage system or a stand-alone extended storage system.

The share system 110 which is shared by the plurality of computer systems has, as a means for dividing and managing the memory in the share system, an identifier allotting unit 210 for recording information as to which systems the individual identifiers are allotted to, a share memory managing unit 211 for managing memory areas on an address space of the global memory 212, that is, division units in the memory in conformity with addresses by using identifiers, and the global memory 212 serving as a location where actual data is arranged.

The identifier allotting unit 210 manages allotment of identifiers in a table format having items "identifier (ID)" and "allotting system name". It is recorded in the identifier allotting unit 210 that, for example, an ID='000001' is allotted to the "SYS A" representative of the processor apparatus 100 and the "SYS B" representative of the processor apparatus 120, an ID='000002' is allotted to the "SYS A" and an ID='000003' is allotted to the "SYS B".

The share memory managing unit 211 manages allotment of the memory areas conforming to identifiers in a table format having items "identifier (ID)", "physical memory address" and "update vector". It is recorded in the share memory managing unit 211 that, for example, an area extending from an address '00000000'X to an address '00000FFF'X of the global memory 212 is managed by an

ID='000001' with the "update vector" being '100'. Although more details of the "update vector" will be described later, it is to be understood that in the "update vector" of 3 bits, a flag is raised in correspondence to three columns in "allotting system name" of the identifier allotting unit 210. More particularly, in an "update vector"='100' corresponding to the ID='000001', '1' corresponds to "SYS A", the middle '0' of '100' corresponds to "SYS B" and the right end '0' of '100' corresponds to "-" indicative of the absence of allotting system. Since the bit is raised ('1') for the "SYS A", it is indicated that data in a memory area managed by the ID='000001' is updated by the processor apparatus 100 (SYS A).

Similarly, it is recorded in the share memory managing unit 211 that an area extending from an address '00001000'X to an address '00001FFF'X of the global memory 212 is managed by an ID='000002' with the "update vector" being '000' to indicate that this area is not updated by any processor apparatus, an area extending from an address '00002000'X to an address '0000FFFF'X of the global memory 212 is managed by an ID='000003' with the "update vector" being '010' to indicate that this area is updated by the processor apparatus 120 (SYS B) and an area extending from an address '00100000'X to an address '001FFFFF'X of the global memory 212 is managed by an ID='000004' with the "update vector" being '000' to indicate that this area is not updated by any processor apparatus.

The memory areas managed by the ID's '000001', ..., '000004' are those depicted in the global memory 212.

Referring now to FIG. 2, the procedure of looking up the inter-system share memory area, which is defined in common to the plurality of systems, during lock request will be described.

FIG. 2 is a flow chart showing the procedure of the lock requesting process in the data sharing method according to the embodiment of the present invention.

The lock requesting process will be described hereunder in accordance with FIG. 1 by following respective steps shown in FIG. 2. It is now assumed that nothing is recorded in columns of the "allotting system name" in the identifier allotting unit in FIG. 1 and columns of the "update vector" in the share memory managing unit 211 are all '000'.

In step 1 of FIG. 2, a program executed by the processor apparatus 100 issues a lock request in accordance with a program executed by the processor apparatus 120 by using a preset ID='000001' in order to access, in a share memory area to be shared by the systems, an area of a length of '1000'X extending from an address '10002000'X of local memory 206 on a precedently reserved address area.

In step 2 of FIG. 2, the inter-system exclusive state conditioning unit 202 of the processor apparatus 100 obtains a locked resource by using the designated identifier. More particularly, the inter-system exclusive state conditioning unit 202 places the ID='000001' in exclusive state. Thus, a request destined for the ID='000001' from another processor apparatus is not acknowledged but is awaited for acknowledgement.

Next, in step 3, the update state confirming unit 203 of the processor apparatus 100 decides whether the designated identifier has already been updated by a different system. More specifically, the update state confirming unit 203 checks whether the corresponding memory area of the global memory 212 on share system 110 which is in correspondence to the designated identifier has been accessed by the different system. If updated, the program proceeds to step 4 but if not updated, the program proceeds to step 13. Steps 4 and 5 will be described later.

5

Here, on the assumption that access to the ID='000001' from the processor apparatus 100 is initial, the program proceeds to the step 13.

In the step 13, the processor apparatus 100 sends to the share system 110 a request that the share system 110 should reserve a memory area of a length of '1000'X for the ID='000001'. The identifier allotting unit 210 of the share system 110 makes a record of the fact that the request for allotment to the ID='000001' is issued from the processor apparatus 100 (SYS A) and the share memory managing unit 211 reserves the global memory 212 on the share system 110 by a '1000'X length memory area corresponding to the ID='000001'.

Next, in step 6, the processor apparatus 100 completes the lock request without transferring data because the access is initial in the lock request process on the processor apparatus 100 and the 'update vector' has not been updated by the different system.

Through the above lock request process, the program on the processor apparatus 100 is allowed to access the area which is in correspondence to the identifier. The program of the processor apparatus 100 is also allowed to lock up/update this area through the procedure similar to that for accessing an area in the system of its own without using any special instructions.

According to the present embodiment, the area can easily be looked up/updated through the procedure similar to that for accessing an area in the system of its own without using any special instructions.

Next, the procedure of requesting unlock of the inter-system share memory area which is defined in common to the plurality of systems will be described with reference to FIG. 3.

FIG. 3 is a flow chart showing the procedure of the unlock requesting process in the data sharing method according to the embodiment of the present invention.

In step 7 of FIG. 3, a program on the processor apparatus 100 designates the ID='000001' and issues an unlock request when access to the share area is completed.

In step 8, the processor apparatus 100 decides whether an update command has been present. In the presence of the update command, the program proceeds to step 9 but in the absence of the update command, the program jumps to step 11. When the program on the processor apparatus 100 only looks up the memory area, update is not commanded.

When access to the ID='000001' from the program on the processor apparatus 100 is initial, no data is formed on the global memory 212 and therefore, the processor apparatus 100 commands update.

Next, in the step 9, data is transferred from a memory area of local memory 206 corresponding to the identifier to the global memory 212. More particularly, the data transferring unit 204 transfers the contents of addresses '10002000'X to '10002FFF'X of local memory 206 corresponding to the ID='000001' to addresses '00000000'X to '00000FFF'X on the global memory 212.

Subsequently, in step 10, an update vector is set. More specifically, the update vector setting unit 205 changes setting of a bit of the update vector managed by the share memory managing unit 211 from '0'B to '1'B to enable the different system (or processor apparatus) to recognize the completion of update by its own system (or processor apparatus). Here, setting of the "update vector" corresponding to the ID='000001' of the share memory managing unit 211 is changed from '000' to '100'. Since the first allotment

6

is made to the "SYS A" representative of the processor apparatus 100 as will be seen from the record in columns in "allotting system name" of the identifier allotting unit 210, setting of the most significant bit of the update vector is changed from '0'B to '1'B.

Next, in the step 11, the inter-system exclusive state conditioning unit 202 releases the locked resource due to the designated identifier. In other words, the exclusive state of the ID='000001' is released.

In step 12, the program on the processor apparatus 100 completes the unlock request.

In the absence of the update command in the step 8, data on the local memory 206 is not transferred onto the global memory 212 and in the step 11, the inter-system exclusive state conditioning unit 202 executes release of the inter-system locked resource exclusive state due to the designated identifier, and in the step 12, the program ends the unlock requesting process.

Next, the lock requesting process when the different system, for example, the processor apparatus 120 looks up/updates a share memory area allotted by the processor apparatus 100 as described above will be described by also making reference to FIG. 2.

In step 1 of FIG. 2, a program on the processor apparatus 120 issues a lock request in accordance with a program on the processor apparatus 100 and a preset ID='000001' in order to access, as a memory area to be shared by the systems, an area of a length '1000'X extending from a precedent reserved address '0E001000'X of local memory 226 on the address space.

In step 2, the inter-system exclusive state conditioning unit 222 of the processor apparatus 120 obtains a locked resource by using a designated identifier. In other words, the inter-system exclusive state conditioning unit 222 places the ID='000001' in exclusive state.

Next, in step 3, the update state confirming unit 223 of the processor apparatus 120 decides whether the designated identifier has already been undated by a different system. Here, since the update state confirming unit 223 knows that the ID='000001' has already been accessed by the processor apparatus 100 by checking the contents of the update vector managed by the share memory managing unit 211, the program proceeds to step 4.

In the step 4, the data transferring unit 224 transfers data from a memory area of global memory 212 corresponding to the identifier to the local memory 226. More particularly, the data transferring unit 224 transfers the contents of addresses '00000000'X to '00000FFF'X of global memory 212 corresponding to the ID='000001' to addresses '0E001000'X to '0E001FFF'X of the local memory 226 on the processor apparatus 120. Through this data transferring process, data corresponding to the ID='000001' on the processor apparatus 100 coincides with data corresponding to the ID='000001' on the processor apparatus 120.

In step 5, the update vector setting unit 225 of the processor apparatus 120 resets the update vector. More specifically, in order to prevent data from being again read on the processor apparatus 120 after transfer of the data, the update vector setting unit 225 resets a bit of the "update vector" from '1'B to '0'B, where B indicates a number in binary. As a result, the "update vector" corresponding to the ID='000001' is reset from '100' to '000'.

In step 6, the lock request is completed and the program is returned to the request program originator. Then, the program on processor apparatus 120 which has completed

the lock request executes look-up/update of data on local memory 226 in correspondence to the ID='000001'.

Next, the unlock requesting process shown in FIG. 3 is executed.

In step 7, an unlock request is issued and in step 8, the update command is decided as to whether to be present. When the look-up/update execution is directed to only look-up, no update command is issued and the locked resource is released in step 11 and the unlock process is completed in step 12.

When the update command is determined to be present in the step 8, data is transferred from the local memory 226 to the global memory 212 in step 9, the update vector is set in step 10, the exclusive state is released in the step 11 and the unlock process ends in the step 12.

By repeating lock→look-up/update→unlock through the processing procedure as above, an area shared by the systems can be accessed. The flow of lock→look-up/update→unlock is the same procedure as that executed when a share area is looked up/updated by all spaces in a system having a plurality of virtual address spaces but differs therefrom in that no real memory control is carried out. The memory shared by the systems can be accessed as easily as the conventional share area.

As described above, according to the present embodiment, the program can easily look up/update an area of the share memory through the procedure similar to that for accessing an area in the system of its own without using any special instructions.

Referring now to FIG. 4, a method will be described in which access operation is carried out by using a group of a plurality of identifiers representative of minimum units of exclusive state.

FIG. 4 schematically shows a composite computer system realizing a data sharing method according to another embodiment of the present invention. Reference numerals identical to those in FIG. 1 designate identical components which are constructed and function identically and will not be detailed.

In the present embodiment, an ID='000001' and an ID='000002' are put together in a group, an ID='000003' and an ID='000004' are put together in another group and these groups are each accessed.

Each of the processor apparatuses 100 and 120 recognizes that the ID='000001' and ID='000002' are put together in one group and it is precedently prescribed in programs on the processor apparatuses 100 and 120 that when a group of the ID='000001' and ID='000002' is accessed, '000001' is used as a representative identifier and when a group of the ID='000003' and ID='000004' is accessed, '000003' is used as a representative identifier.

For example, in the local area allotting unit 201, an area of a length of '1000'X extending from an address '10002000'X to an address '10002FFF'X of the local memory 206 is managed by ID='000001', an area of a length of '1000'X extending from an address '10003000'X to an address '10003FFF'X of the local memory 206 is managed by ID='000002', an area of a length of '1000'X extending from an address '10004000'X to an address '10004FFF'X of the local memory 206 is managed by ID='000003' and an area of a length of '1000'X extending from an address '10005000'X to an address '10005FFF'X is managed by ID='000004'. The memory areas managed by the ID's = '000001', . . . , '000004' are those depicted in the local memory 206.

For example, in the local area allotting unit 221, an area of a length of '1000'X extending from an address '0E003000'X to an address '0E003FFF'X of the local memory 226 is managed by ID='000001', an area of a length of '1000'X extending from an address '0E004000'X to an address '0E004FFF'X of the local memory 226 is managed by ID='000002', an area of a length of '1000'X extending from an address '0E001000'X to an address '0E001FFF'X of the local memory 226 is managed by ID='000003' and an area of a length of '1000'X extending from an address '0E002000'X to an address '0E002FFF'X of the local memory 226 is managed by ID='000004'. The memory areas managed by the ID's = '000001', . . . , '000004' are those depicted in the local memory 226.

Further, it is recorded in the identifier allotting unit 210 that for example, the ID='000001' is allotted to "SYS A" representative of the processor apparatus 100 and "SYS B" representative of the processor apparatus 120, the ID='000002' is allotted to "SYS A" and "SYS B", the ID='000003' is allotted to "SYS B" and "SYS A" and the ID='000004' is allotted to "SYS B" and "SYS A".

Also, it is recorded in the share memory managing unit 211 that for example, an area extending from an address '00000000'X to an address '000000FFFF'X of the global memory 212 is managed by ID='000001' with the "update vector" being '000', an area extending from an address '00006500'X to an address '000074FF'X of the global memory 212 is managed by ID='000002' with the "update vector" being '010', an area extending from an address '00004000'X to an address '00004FFF'X of the global memory 212 is managed by ID='000003' with the "update vector" being '000' and an area extending from an address '00008000'X to an address '00008FFF'X of the global memory 212 is managed by ID='000004' with the "update vector" being '100'.

Referring again to FIG. 2, the process in which a lock request is made by using a group of a plurality of identifiers will be described.

In step 1 of FIG. 2, a program on the processor apparatus 100 issues a lock request in accordance with a program on the processor apparatus 120 and a preset ID='000003' in order to access, as memory areas to be shared by the systems, an area of a length '1000'X extending from a precedently reserved address '10004000'X of local memory 206 on an address space and an area of a length '1000'X extending from an address '10005000'X of the local memory 206. Here, the areas managed by the ID='000003' and ID='000004' are to be locked and in this case, the lock request is made in accordance with a representative identifier.

In step 2, the inter-system exclusive state conditioning unit 202 of the processor apparatus 100 obtains a locked resource by using the designated identifier. In other words, the inter-system exclusive state conditioning unit 202 places the ID='000003' in exclusive state. As a result, the ID='000004' is not placed in exclusive state but, because of the agreement that the ID='000003' and ID='000004' are handled in group by any of the processor apparatuses 100 and 120, a locked resource can substantially be obtained for the ID='000004' without making a request for exclusive state of the ID='000004' when the ID='000003' is placed in exclusive state.

Next, in step 3, the update state confirming unit 203 of the processor apparatus 100 decides whether the designated identifier has already been updated by a different system. More particularly, the update state confirming unit 203

checks whether a memory area of global memory 212 on the share system 110 corresponding to the designated identifier is accessed by the different system.

Even when the lock request by the ID='000003' is issued from the processor apparatus 100, the update state confirming unit 203 does not confirm only the representative identifier but checks both the ID's '000003' and '000004' to confirm the update states. In the state shown in FIG. 4, the update vector for the ID='000003' in the share memory managing unit 211 is '000', indicating that it is not updated but the update vector for the ID='000004' is '100', indicating that it is updated by the processor apparatus 120 (SYS B), so that the program proceeds to step 4.

In the step 4, the data transferring unit 204 transfers data from the global memory 212 to the local memory 206.

In step 5, the update vector setting unit 205 resets the update vector for the ID='000004' in the share memory managing unit 211.

Next, in step 6, the processor apparatus 100 completes the lock request and looks up/updates the read-out data.

Through the above lock requesting process, the program on the processor apparatus 100 is permitted to access the plurality of areas corresponding to the identifier. Even when a plurality of units (areas) to be looked up/updated are provided, the overhead due to the exclusive state is incurred only once, proving that it can be reduced.

The program can look up/update the area easily through the procedure similar to that for accessing the area in the system of its own without using any special instructions.

The process for unlock request is the same as that described in connection with FIG. 3. Since only a locked resource is obtained in conformity with a representative identifier, the locked resource is released in step 11 in conformity with only the identifier for which the locked resource is obtained.

According to the present embodiment, the area can be looked up/updated easily through the procedure similar to that for accessing the area in the system of its own without using any special instructions.

In addition, the overhead due to the exclusive state incurred when locked resources are obtained for a plurality of areas can be reduced.

Referring now to FIG. 5, a method of making the access to the share system asynchronous to the access to the local memory when a program on the processor apparatus looks up/updates the memory shared by the systems will be described.

FIG. 5 schematically shows a composite computer system realizing a data sharing method according to still another embodiment of the present invention. Reference numerals identical to those in FIG. 1 designate identical components which are constructed and function identically and will not be detailed.

On the processor apparatus 100, a program A 207 is operated as a program which operates while looking up/updating the local memory 206 in correspondence to the memory shared by the systems.

It is now assumed that allotment of areas of local memory 206 managed by a local area allotting unit (not shown) in the processor apparatus 100 is similar to that shown in FIG. 4.

On the processor apparatus 120, a program B 227 is operated as a program which operates while looking up/updating the local memory 226 in correspondence to the memory shared by the systems.

The allotment of an area of local memory 226 managed by the local area allotting unit in the processor apparatus 120 is similar to that shown in FIG. 4.

Further, the allotting system name in the identifier allotting unit 210 is similar to that shown in FIG. 4.

Also, the managing condition of the global memory 212 serving as the share memory in the share memory managing unit 211 is similar to that shown in FIG. 4.

Data shared by the program A 207 and program B 227 is unlike user data whose integrity must be ensured throughout all the systems, and includes, for example, busy rates of the CPU's on the processor apparatuses 100 and 120 and operating information of the share resource which is allowed to have a bit of error.

When the data as above is shared, interchange of data occurs periodically and therefore, the overhead due to accessing the share system 110 and the overhead due to the exclusive state are desired to be reduced as far as possible. Accordingly, the lock/unlock request to the share system 110 is not operated on an extension of the operation of each program but data access to the share system 110 is carried out at a period of relatively reduced overhead in accordance with a totally different program to thereby reflect data upon the local memories 206 and 226.

The program A 207 and program B 227 are operated at a timing different from the timing at which data is reflected upon the local memories 206 and 226 and only the local memories 206 and 226 are looked up/updated as necessary.

By making the timing of access to the local memory different from the timing of access to the global memory to perform the asynchronous and non-interlocked process in this manner, the overhead for the shared system can be reduced.

According to the present embodiment, the area can be looked up/updated easily through the procedure similar to that for accessing the area in the system of its own without using any special instructions.

Further, by making the timing at which data is looked up/updated on each computer system different from the timing at which data on the share system is reflected upon the computer system to perform the asynchronous and non-interlocked process, the overhead for the share system can be reduced.

According to the present invention, when the memory is shared by a plurality of computer systems, the area can be looked up/updated through the procedure similar to that for accessing the share area in the conventional system.

We claim:

1. A data sharing method in which data used in common for processor apparatuses of a plurality of computer systems is shared by a share memory in a share system, comprising the steps of:

dividing said share memory in said share system into constant units and causing said share system to control the length of the individual division units by using identifiers which are definitely determined as viewed from said plurality of computer systems and said share system;

when a processor apparatus designates a desired area on an address space in its computer system by using an identifier to obtain a locked resource, causing said processor apparatus to check whether data in a corresponding unit in said share memory is updated and to transfer data in the corresponding unit in said share memory to said address space when the data is updated; and

when said processor apparatus releases the locked resource, causing said processor apparatus to transfer

11

the data to the corresponding unit in said share memory if said data in said address space is updated.

2. A data sharing method according to claim 1, wherein in said computer system, said processor apparatus accesses a group of a plurality of identifiers and is caused to obtain a locked resource in said share system by using a representative identifier of the identifier group. 5

3. A data sharing method according to claim 1, wherein in said computer system, the access timing at which said processor apparatus looks-up/updates data on said computer system is made to be asynchronous to the access timing at which said processor apparatus reflects data upon said share system on said computer system. 10

4. A data sharing system in which data used in common for processor apparatuses of a plurality of computer systems is shared by a share memory in a share system, comprising: 15

said share system having its share memory divided into constant units, for managing the individual division units by using identifiers which are definitely determined; and 20

said processor apparatus coupled to said share system and being operative, when designating a desired area on an address space in said computer system by using an

12

identifier to obtain a locked resource, to check whether data in a corresponding unit in said share memory is updated and to transfer the data in the corresponding unit in said share memory to said address space when the data is updated,

wherein when releasing the locked resource, said processor apparatus transfers the data to the corresponding unit in said share memory if the data in said address space is updated.

5. A data sharing system according to claim 4, wherein in said computer system, said processor apparatus accesses a group of a plurality of identifiers and obtains a locked resource in said share system by using a representative identifier of the identifier group. 15

6. A data sharing system according to claim 4, wherein in said computer system, the access timing at which said processor apparatus looks-up/updates data on said computer system is made to be asynchronous to the access timing at which said processor apparatus reflects data upon said share system on said computer system. 20

* * * * *